Tehnici Web
CURSUL 11

Semestrul I, 2024-2025

Carmen Chirita

SVG - Scalable Vector Graphics

https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial

- limbaj de marcare bazat pe XML pentru descrierea de elemente grafice vectoriale 2D

- componentele principale ale imaginii sunt figuri geometrice (dreptunghi, cerc) și curbe (simple si complexe) care sunt descrise matematic; aceasta asigura independenta de rezolutie, scalabilitatea

- imaginile SVG sunt definite în fișiere XML și pot fi randate la orice dimensiune fără pierderea calității

SVG

- fisierele SVG pot fi editate si cu editoare specializate (Adobe Illustrator, Inkscape), dar si ca fisiere XML

- fisierele SVG pot fi incluse in fisierele HTML

```
<img src="image.svg" alt="svg image">
<iframe src="image.svg"></iframe>
<object data="image.svg" type="image/svg+xml"></object>
```

- imaginile (codul) SVG pot fi incluse direct in fisiere HTML

```
<svg  width="450" height="500"   id="elsvg"
 xmlns="http://www.w3.org/2000/svg">
          .........
</svg>
```

Sintaxa

- fiind bazat pe XML, SVG e case-sensitive: atenție la majuscule în cazul elementelor și al atributelor

- valorile atributelor se scriu între ghilimele, chiar și cele numerice

Exemplu

```
<body>
<h1>SVG inclus in HTML</h1>

<svg width="100" height="100" xmlns="http://www.w3.org/2000/svg">
//container pentru grafica
   <circle cx="50" cy="50" r="40" stroke="red" stroke-width="5" fill="yellow" />
</svg>

</body>
```

SVG inclus in HTML

Atribute specifice: cx, cy, r //coordonatele centrului și raza
Atributul fill //culoarea continutului
Atributul stroke //culoarea conturului
Atributul stroke-width //grosimea conturului

Elemente SVG predefinite

<rect>

<circle>

<ellipse>

<line>

<polyline>

<polygon>

<path>

<text>

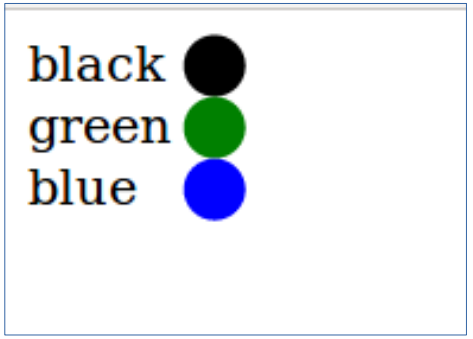
Elementele SVG <defs>, <use>

SVG permite ca obiectele grafice să fie definite pentru o reutilizare ulterioară.

Elementele de referință se vor defini în interiorul unui element <defs> și pot fi referite cu <use>.

De exemplu, <defs> poate fi utilizat la crearea gradientilor

```
<svg width="500" height="600">

 <defs>

    <circle id="port" cx="10" cy="0" r="10"/>

 </defs>

  <text y="15">black</text>
  <use x="50" y="10" href="#port" style="fill: black;"/>
  <text y="35">green</text>
  <use x="50" y="30" href="#port" style="fill: green;"/>
  <text y="55">blue</text>
  <use x="50" y="50" href="#port" style="fill: blue;"/>


</svg>
```



SVG Rectangle - <rect>

utilizat pentru a crea un dreptunghi și variații ale unei forme de dreptunghi

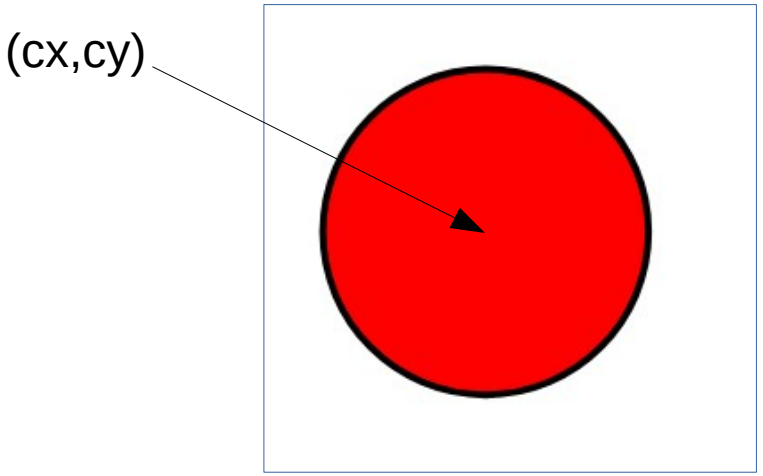
```
<body>

<svg width="400" height="200" xmlns="http://www.w3.org/2000/svg">
  <rect width="200" height="100" fill=”green”  stroke-width=”3”  stroke=”red" />
</svg>

</body>
```

Atribute specifice: width , height, x, y
Atributele rx și ry //colturi rotunjite
Atributul fill //culoarea continutului
Atributul stroke //culoarea conturului
Atributul stroke-width //grosimea conturului

Proprietati CSS:
fill: culoarea continutului
stroke: culoarea conturului
stroke-width: grosimea conturului
fill-opacity:0-1, stroke-opacity:0-1,opacity:0-1

SVG Circle - <circle> (deseneaza un cerc)

```
<circle cx="100" cy="100" r="70"
 style="stroke:black; stroke-width:3; fill:red" />
```



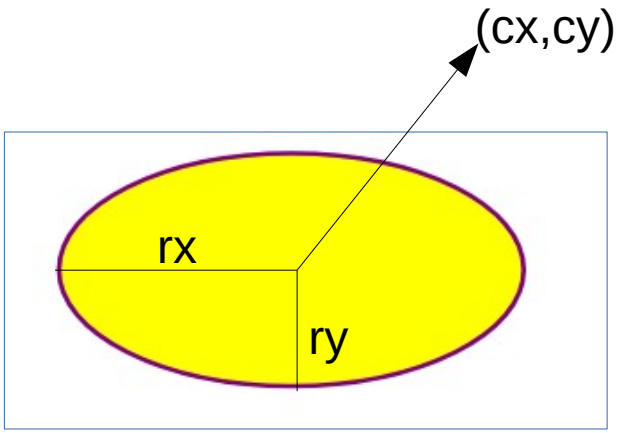
Atribute specifice:
cx, cy //coordonatele centrului
r //raza cercului

Proprietati CSS:

fill: culoarea continutului
stroke: culoarea conturului
stroke-width: grosimea conturului
fill-opacity:0-1, stroke-opacity:0-1,opacity:0-1

SVG Ellipse - <ellipse> (deseneaza o elipsa)

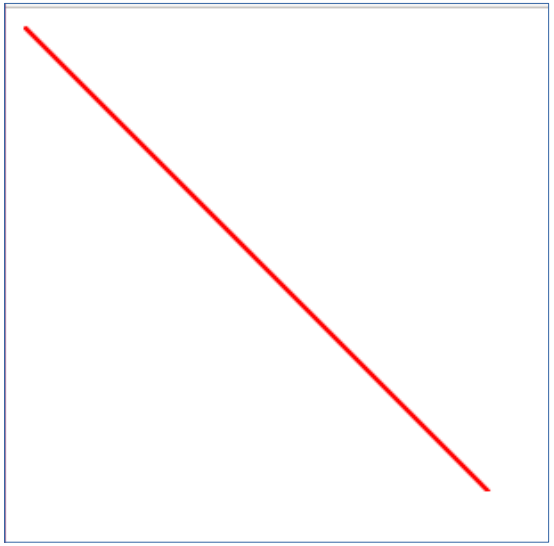
```
<ellipse cx="200" cy="80" rx="100" ry="50"
 fill="yellow" stroke="purple" stroke-width="2" />
```



Atribute specifice:
cx, cy //coordonatele centrului elipsei
rx //raza orizontala
ry //raza verticala

SVG Line - <line> (deseneaza o linie)

```
<line x1="0" y1="0" x2="200" y2="200"
stroke="rgb(255,0,0)" stroke-width="2" />
```

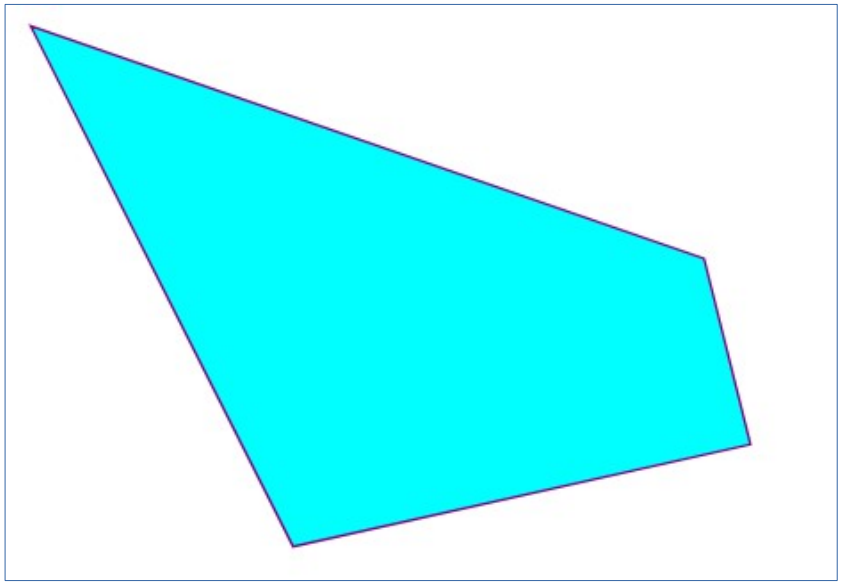


Atribute specifice:
x1, y1 //coordonatele de inceput
x2, y2 //coordonatele de sfârșit

SVG Polygon - <polygon>

este utilizat pentru a crea o imagine care conține cel puțin trei laturi.

```
<polygon points="10,10 300,110 320,190 123,234"
fill="cyan" stroke="purple" stroke-width="1" />
```

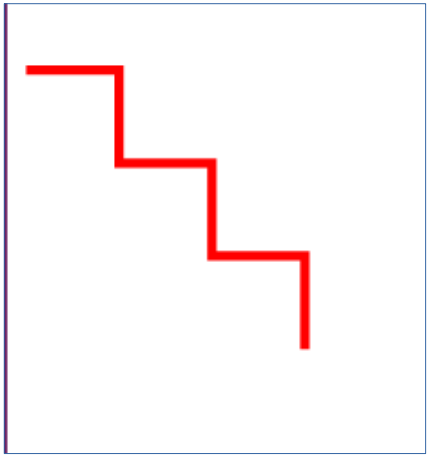


Atribute specifice:
points : //coordonatele varfurilor

SVG Polyline - <polyline>

este utilizat pentru a crea orice formă care constă numai din linii drepte

```
<polyline points="0,40 40,40 40,80 80,80 80,120 120,120 120,160" fill="white" stroke="red" stroke-width="4" />
```

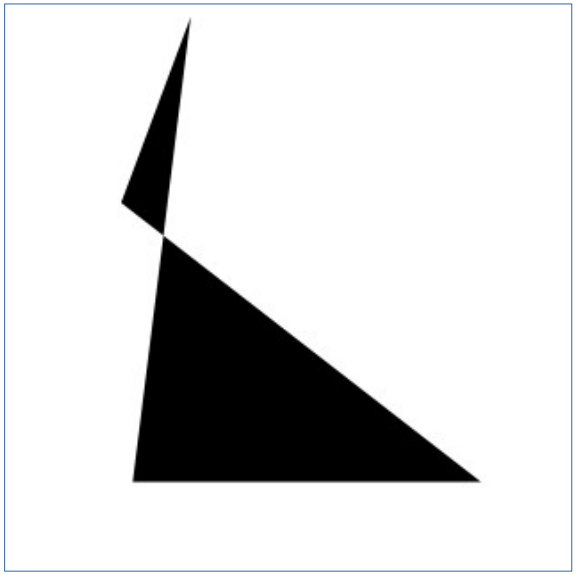


Atribute specifice:
points : //coordonatele varfurilor

SVG Path - <path>

este utilizat pentru a crea forme complexe combinand linii, arcuri, curbe, etc.

```
<path d="M100 0 L75 200 L225 200 L70 80 Z" />
```



Comenzi

M = moveto

L = lineto

Z = closepath

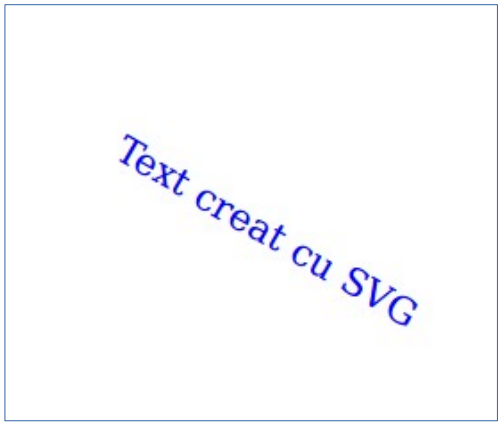
H = horizontal lineto

V = vertical lineto

A = elliptical Arc

SVG Text - <text> (defineste un text)

```
<text x="50" y="50" fill="blue" transform="rotate(30 20,40)">Text creat cu SVG</text>
```

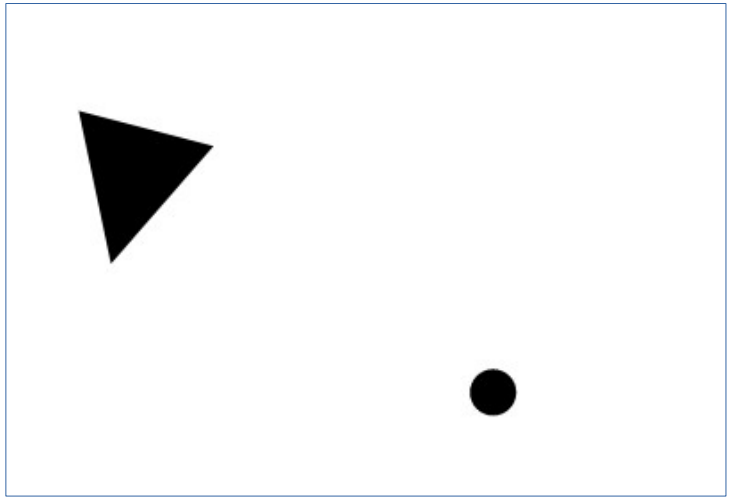


Atribute specifice:
x,y : //coordonatele de inceput

Animatie SVG

<animate>
<animateTransform>
<animateMotion>

```
<polygon points="60,30 90,90 30,90">
      <animateTransform attributeName="transform"
                        type="rotate"
                        from="0 60 70"
                        to="360 60 70"
                        dur="10s"
                        repeatCount="4"/>
   </polygon>
```



```
<circle cx="230" cy="32" r="10" >
<animateMotion
      path= "M0,0 L50,132 L-50, 132 L0,0"
       begin="3s" dur="10s"
       repeatCount="indefinite" />
</circle>
```

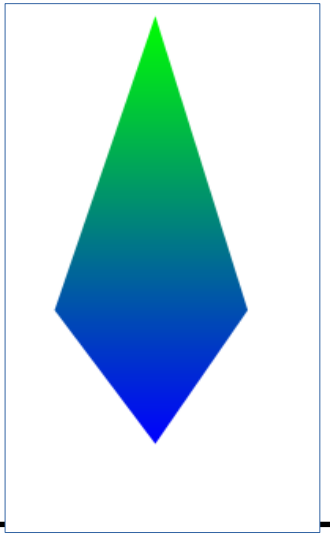
svg.html

Gradienti in SVG- <linearGradient>

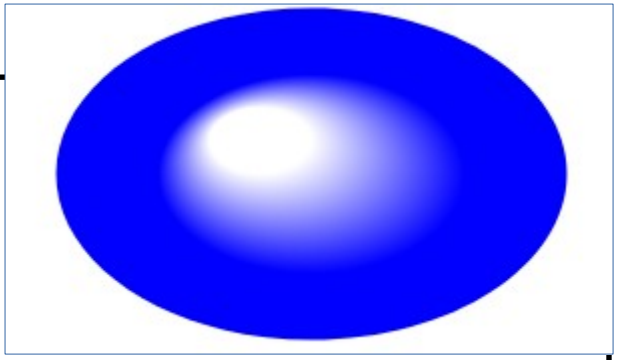
```
<svg>

<defs> //defineste elemente SVG care vor
fi utilizate ulterior
<linearGradient id="myLG“
   x1="0%" y1="0%"   x2="0%" y2="100%“
spreadMethod="pad">
<stop offset="0%"   stop-color="#00ff00" stop-opacity="1"/>
<stop offset="100%" stop-color="#0000ff" stop-opacity="1"/>
</linearGradient>
</defs>

<path d="M 230,32  L181,175 L 230,240
                  L 275,175 L 230,32 "
     style="fill:url(#myLG)“ />
</svg>
```

Gradienti:

Pe verticala: x1 = x2
Pe orizontala: y1 = y2
Unghiulari: x1≠x2, y1≠y2



Gradienti in SVG -<radialGradient>



```
<svg height="150" width="500">

 <defs>

    <radialGradient id="grad1" cx="50%" cy="50%" r="30%"
fr="10%" fx="40%" fy="40%>
      <stop offset="0%" style="stop-color:rgb(255,255,255);
stop-opacity:0" />
      <stop offset="100%" style="stop-color:rgb(0,0,255);
stop-opacity:1" />
    </radialGradient>

  </defs>

  <ellipse cx="200" cy="70" rx="85" ry="55" fill="url(#grad1)" />

</svg>
```

CANVAS

Elementul HTML <canvas> este folosit pentru a desena grafica, cu ajutorul JavaScript.

Elementul <canvas> este doar un container pentru grafică (este necesar un script pentru a desena grafica).

Canvas are mai multe metode pentru a desena linii, figuri, cercuri, text și inserare de imagini.

CANVAS

```
#canvas {border: 2px solid black
body {background-color: red;}
```

```
Suprafața de desen:

<canvas id=“canvas” width=“500” height=“400”>
 Continut alternativ </canvas>
```

```
canvas = document.getElementById("canvas");
context = canvas.getContext("2d");
```

https://developer.mozilla.org/en-US/docs/Web/API/HTMLCanvasElement

http://www.w3.org/TR/2010/WD-html5-20100624/the-canvas-element.html

HTMLCanvasElement metoda toDataUrl

Returnează un URL de date ce conține o reprezentare a imaginii în formatul specificat de parametrul de tip (implicit png).

canvas.toDataURL(type, encoderOptions)

```
// type poate fi: image/png // “image/jpeg”, “image/webp”
//encoderOptions: calitatea imaginii (intre 0-1)
```

```
<body>
<div id="imgcanvas">
 <h1>Imagine creata cu canvas </h1>
 </div>
</body>
```

Canvas-ul poate fi folosit ca o imagine bitmap

```
var url = canvas.toDataURL();
 var newImg = document.createElement("img");
 newImg.src = url;
 var newImgParent = document.getElementById("imgcanvas");
 newImgParent.appendChild(newImg);
```

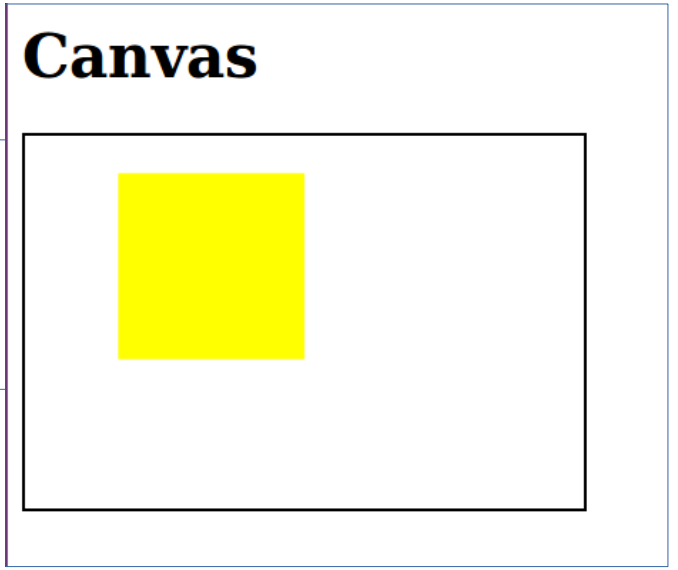
Canvas: fillStyle și fillRect()

fillStyle //stilul folosit în interiorul formelor
poate fi culoare /gradient/ patern
fillRect(x,y,width,height) //deseneaza un dreptunghi

```
var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");

ctx.fillStyle = "yellow";
ctx.fillRect(50, 20, 100, 100);
```

```
<style type="text/css" >
#canvas {border: 2px solid black;}
</style>
```

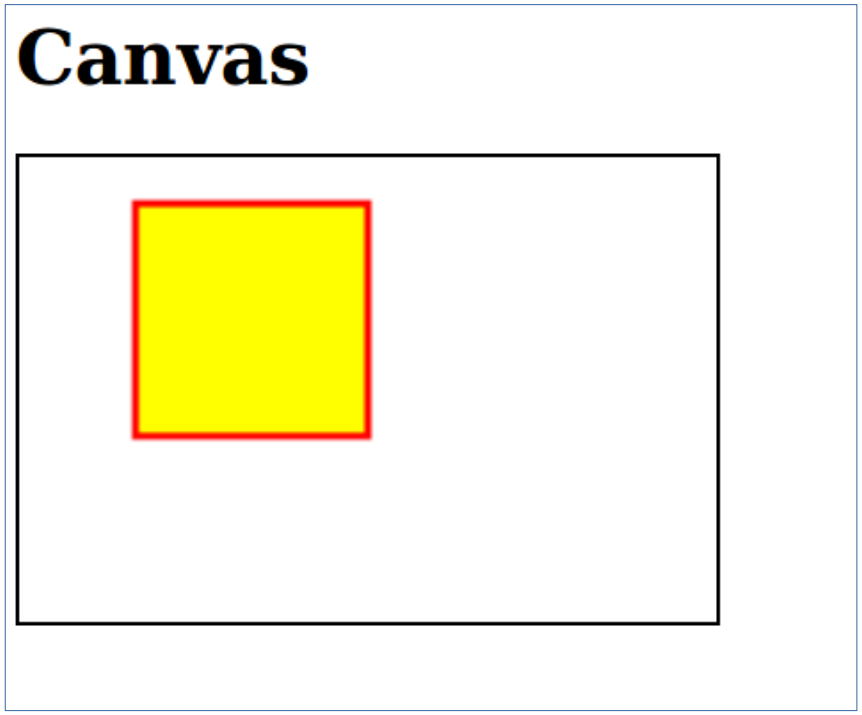
```
<body>
<h1> Canvas </h1>
<canvas id="canvas" width="300" height="200"> alt</canvas>
</body>
```



Stroke (conturul) si Fill (continutul)

```
ctx.strokeStyle = "red";
ctx.lineWidth = 3;
ctx.strokeRect(50, 20, 100, 100);

ctx.fillStyle = "yellow";
ctx.fillRect(50, 20, 100, 100);
```

strokeStyle
lineWidth
fillStyle



strokeRect(x,y,width,height)
fillRect(x,y,widht,height)
clearRect(x,y,widht,height)

(x,y) coltul stanga sus

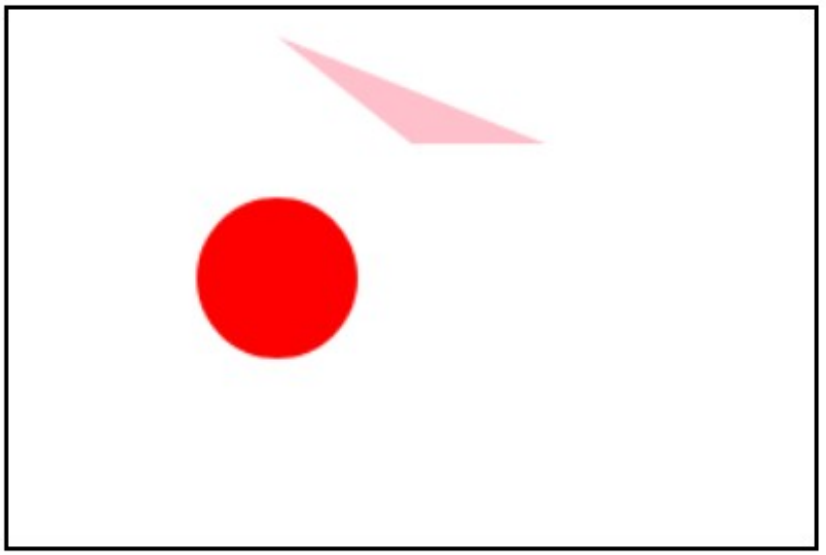
Desenarea figurilor

beginPath() //incepe desenarea unei noi figuri
closePath() //inchide figura de la punctul curent
la punctul de plecare

Desenarea unei linii
moveTo(x,y) //coordonatele de inceput
lineTo(x,y) // coordonatele de sfarsit
lineCap //stilul capetelor liniei ("butt|round|square")
stroke() //deseneaza efectiv linia

Desenarea unei cerc
beginPath()
arc(x,y,r,startangle,endangle, counterclockwise)
//creeaza un cerc (arc de cerc)
stroke() /* fill() //deseneaza doar conturul sau
coloreaza si continutul

```
var canvas=document.getElementById("canvas");
var ctx = canvas.getContext("2d");

ctx.beginPath();
ctx.moveTo(100,10);
ctx.lineTo(150,50);
ctx.lineTo(200,50);
ctx.closePath();
ctx.lineCap = "round";
ctx.fillStyle = "pink";
ctx.fill();

ctx.beginPath();
ctx.arc(100, 100, 30,  0, 2* Math.PI);
ctx.fillStyle = "red";
ctx.fill();
ctx.closePath();
```

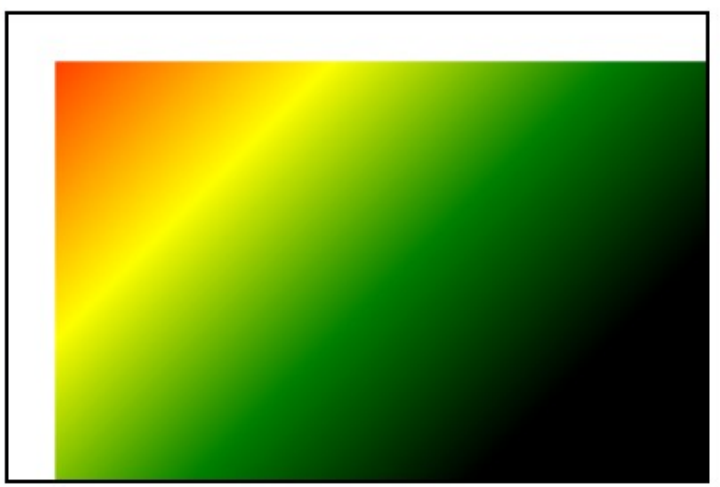


```
<body>
<h1> Canvas </h1>
<canvas id="canvas" width="300" height="200"> alt</canvas>
</body>
```

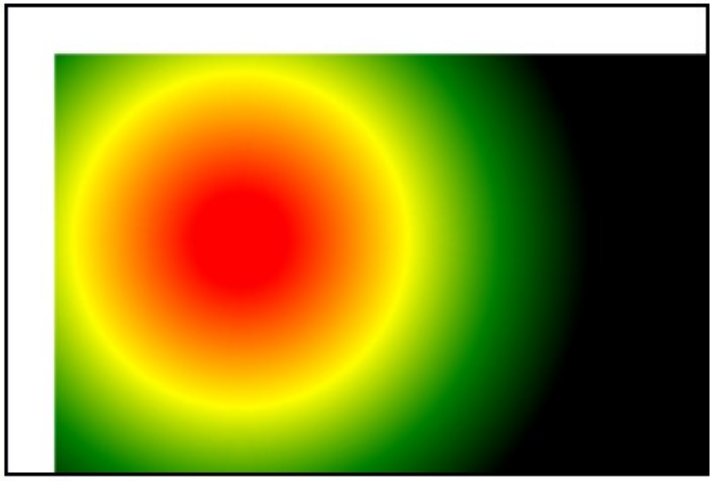
linearGadient, radialGradient

createLinearGradient(x0,y0,x1,y1)
createRadialGradient(x0,y0,r0,x1,y1,r1)
addColorStop()

```
var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");

var fade = ctx.createLinearGradient(0, 0, 200, 200);
fade.addColorStop(0, "red");
fade.addColorStop(0.4, "yellow");
fade.addColorStop(0.7,"green");
fade.addColorStop(1.0, "black");

ctx.fillStyle = fade;
ctx.fillRect(20, 20, 300, 300);
```



```
var fade = ctx.createRadialGradient(100,100,20,100,100,150);
```



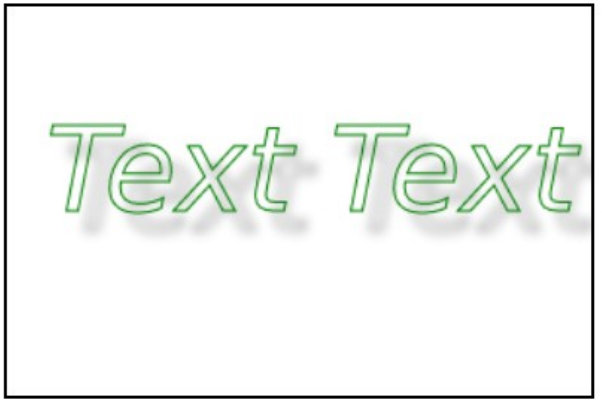
Text și shadow

font
strokeText(text,x,y)
fillText(text,x,y)

shadowOffsetX
shadowOffsetY
shadowBlur
shadowColor

```
ctx.shadowOffsetX = 10;
ctx.shadowOffsetY = 10;
ctx.shadowBlur    = 4;
ctx.shadowColor   = "#666666";
ctx.fillStyle ="green";
//ctx.strokeStyle = "green";

ctx.font = "italic 60px sans-serif";
ctx.lineWidth = 1;
ctx.fillText("Text Text",20, 105);
//ctc.strokeText("Text Text",20, 105);
```

fill text



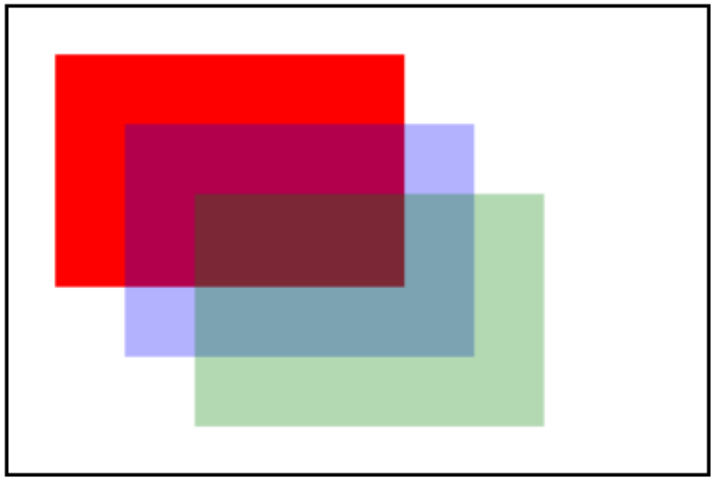
stroke text

CANVAS: globalAlpha si globalCompositeOperation

globalAlpha()
transparenta (intre 0-1=opac)

globalCompositeOperation
modul de amestecare a culorilor

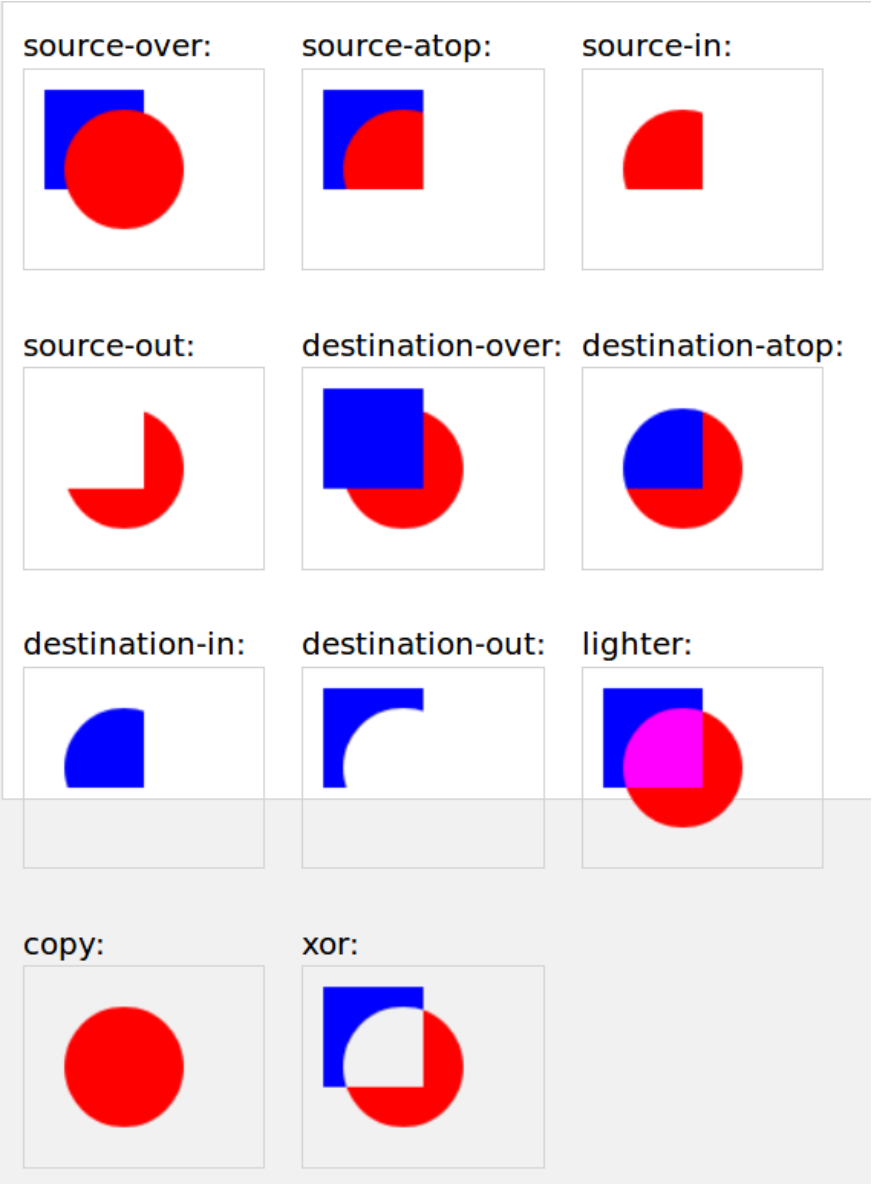
```
ctx.fillStyle = "red";
ctx.fillRect(20, 20, 150, 100);

ctx.globalAlpha = 0.3;
ctx.fillStyle = "blue";
ctx.fillRect(50, 50, 150, 100);
ctx.fillStyle = "green";
ctx.fillRect(80, 80, 150, 100);
```



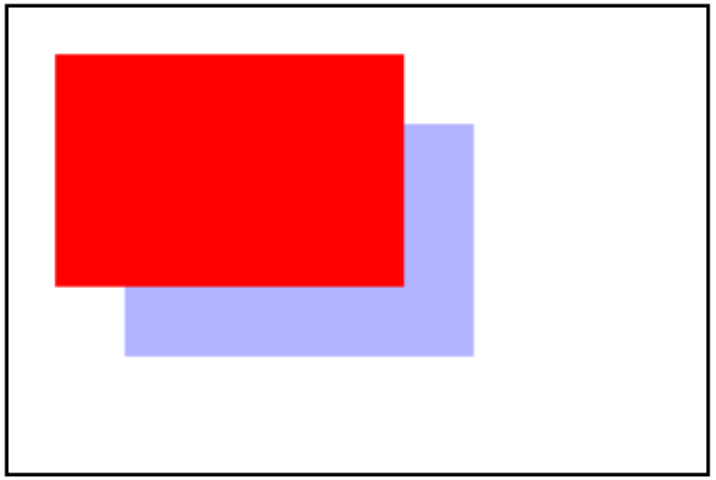
globalCompositeOperation
modul de amestecare a culorilor

specifica modul în care o imagine sursă (nouă)
este plasata pe o imagine destinație (existentă).

implicit:source-over



```
ctx.fillStyle = "red";
ctx.fillRect(20, 20, 150, 100);

ctx.globalAlpha = 0.3;
ctx.globalCompositeOperation="destination-over";
ctx.fillStyle = "blue";
ctx.fillRect(50, 50, 150, 100);
```



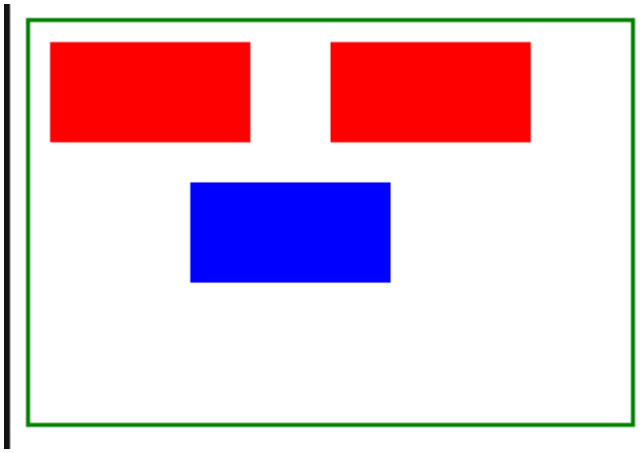
Transformari si salvarea contextului

ctx.save()
ctx.restore()

ctx.translate(x,y)
ctx.rotate(degrees*Math.PI/180)
ctx.scale(factor)

se aplica tuturor elementelor desenate

```
    ctx.fillStyle = 'red';
    ctx.fillRect(10, 10, 100, 50);

    ctx.save();     // salvăm starea curentă

// schimbăm stilul de umplere și translatarea
    ctx.translate(70, 70);
    ctx.fillStyle = 'blue';
    ctx.fillRect(10, 10, 100, 50);

    ctx.restore(); // restauram starea anterioara

    ctx.fillRect(150, 10, 100, 50);
```



Desenarea imaginilor

```
drawImage(image, x, y);
drawImage(image,x,y,width,height);
drawImage(image, sx, sy, sWidth, sHeight, dx, dy,
dWidth, dHeight);
```

```
const img = new Image(); // creez un obiect imagine

img.src = ”myImage.png”; //setez sursa imaginii

img.onload = function()  {

   //desenarea imaginii pe canvas când imaginea este incarcata

    ctx.drawImage(img, 0, 0, canvas.width, canvas.height);

};
```

Desenarea imaginilor

fisier.html

```
.............
<canvas id="canvas" width="300" height="200"> alt</canvas>
.............
```

fisier.js

```
var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");

var image = new Image();
 image.src=”iarna.jpg”;

image.onload=function() {
 ctx.drawImage(image, 0, 30, 150, 90);
}
```

Animatii

Pași pentru desenarea unui frame:

1. curatarea canvasului : putem folosi clearRect()
2. salvarea stării canvasului
3. desenarea figurilor de animat
4. restaurarea stării canvasului

Animatii

Funcții care pot fi folosite în animații:

setInterval();

setTimeout();

animationID **= requestAnimationFrame**(callback);
/* cere browserului să apeleze funcția callback pentru a actualiza o animație înainte de următoarea redesenare */
cancelAnimationFrame(animationID**)**
/*anuleaza o animație care a fost solicitată anterior cu requestAnimationFrame

https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Basic_animations